Digesting Virtual "Geek" Culture: The Summarization of Technical Internet Relay Chats

Liang Zhou and Eduard Hovy
University of Southern California
Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
{liangz, hovy} @isi.edu

Abstract

This paper describes a summarization system for technical chats and emails on the Linux kernel. To reflect the complexity and sophistication of the discussions, they are clustered according to subtopic structure on the sub-message level, and immediate responding pairs are identified through machine learning methods. A resulting summary consists of one or more mini-summaries, each on a subtopic from the discussion.

1 Introduction

The availability of many chat forums reflects the formation of globally dispersed virtual communities. From them we select the very active and growing movement of Open Source Software (OSS) development. Working together in a virtual community in non-collocated environments, OSS developers communicate and collaborate using a wide range of web-based tools including Internet Relay Chat (IRC), electronic mailing lists, and more (Elliott and Scacchi, 2004). In contrast to conventional instant message chats, IRCs convey engaging and focused discussions on collaborative software development. Even though all OSS participants are technically savvy individually, summaries of IRC content are necessary within a virtual organization both as a resource and an organizational memory of activities (Ackerman and Halverson, 2000). They are regularly produced manually by volunteers. These summaries can be used for analyzing the impact of virtual social interactions and virtual organizational culture on software/product development.

The emergence of email thread discussions and chat logs as a major information source has prompted increased interest in thread summarization within the Natural Language Processing (NLP) community. One might assume a smooth transition from text-based summarization to email and chat-based summarizations. However, chat falls in the genre of correspondence, which requires dialogue and conversation analysis. This property makes summarization in this area even more difficult than traditional summarization. In particular, topic "drift" occurs more radically than in written genres, and interpersonal and pragmatic content appears more frequently. Questions about the content and overall organization of the summary must be addressed in a more thorough way for chat and other dialogue summarization systems.

In this paper we present a new system that clusters sub-message segments from correspondences according to topic, identifies the sub-message segment containing the leading issue within the topic, finds immediate responses from other participants, and consequently produces a summary for the entire IRC. Other constructions are possible. One of the two baseline systems described in this paper uses the timeline and dialogue structure to select summary content, and is quite effective. We use the term *chat* loosely in this paper. Input IRCs for our system is a mixture of chats and

Proceedings of the 43rd Annual Meeting of the ACL, pages 298–305,
Ann Arbor, June 2005.

emails that are indistinguishable in format observed from the downloaded corpus (Section 3).

In the following sections, we summarize previous work, describe the email/chat data, intra-message clustering and summary extraction process, and discuss the results and future work.

2 Previous and Related Work

There are at least two ways of organizing dialogue summaries: by dialogue structure and by topic.

Newman and Blitzer (2002) describe methods for summarizing archived newsgroup conversations by clustering messages into subtopic groups and extracting top-ranked sentences per subtopic group based on the intrinsic scores of position in the cluster and lexical centrality. Due to the technical nature of our working corpus, we had to handle intra-message topic shifts, in which the author of a message raises or responds to multiple issues in the same message. This requires that our clustering component be not message-based but sub-message-based.

Lam et al. (2002) employ an existing summarizer for single documents using preprocessed email messages and context information from previous emails in the thread.

Rambow et al. (2004) show that sentence extraction techniques are applicable to summarizing email threads, but only with added email-specific features. Wan and McKeown (2004) introduce a system that creates overview summaries for ongoing decision-making email exchanges by first detecting the issue being discussed and then extracting the response to the issue. Both systems use a corpus that, on average, contains 190 words and 3.25 messages per thread, much shorter than the ones in our collection.

Galley et al. (2004) describe a system that identifies agreement and disagreement occurring in human-to-human multi-party conversations. They utilize an important concept from conversational analysis, adjacent pairs (AP), which consists of initiating and responding utterances from different speakers. Identifying APs is also required by our research to find correspondences from different chat participants.

In automatic summarization of spoken dialogues, Zechner (2001) presents an approach to obtain extractive summaries for multi-party dialogues in unrestricted domains by addressing intrinsic issues specific to speech transcripts. Automatic question detection is also deemed important in this work. A decision-tree classifier was trained on question-triggering words to detect questions among speech acts (sentences). A search heuristic procedure then finds the corresponding answers. Ries (2001) shows how to use keyword repetition, speaker initiative and speaking style to achieve topical segmentation of spontaneous dialogues.

3 Technical Internet Relay Chats

GNUe, a meta-project of the GNU project[1]–one of the most famous free/open source software projects–is the case study used in (Elliott and Scacchi, 2004) in support of the claim that, even in virtual organizations, there is still the need for successful conflict management in order to maintain order and stability.

The GNUe IRC archive is uniquely suited for our experimental purpose because each IRC chat log has a companion summary digest written by project participants as part of their contribution to the community. This manual summary constitutes gold-standard data for evaluation.

3.1 Kernel Traffic[2]

Kernel Traffic is a collection of summary digests of discussions on GNUe development. Each digest summarizes IRC logs and/or email messages (later referred to as chat logs) for a period of up to two weeks. A nice feature is that direct quotes and hyperlinks are part of the summary. Each digest is an extractive overview of facts, plus the author's dramatic and humorous interpretations.

3.2 Corpus Download

The complete Linux Kernel Archive (LKA) consists of two separate downloads. The Kernel Traffic (summary digests) are in XML format and were downloaded by crawling the Kernel Traffic site. The Linux Kernel Archives (individual IRC chat logs) are downloaded from the archive site. We matched the summaries with their respective chat logs based on subject line and publication dates.

3.3 Observation on Chat Logs

[1] http://www.gnu.org
[2] http://kt.hoser.ca/kernel-traffic/index.html

Upon initial examination of the chat logs, we found that many conventional assumptions about chats in general do not apply. For example, in most instant-message chats, each exchange usually consists of a small number of words in several sentences. Due to the technical nature of GNUe, half of the chat logs contain in-depth discussions with lengthy messages. One message might ask and answer several questions, discuss many topics in detail, and make further comments. This property, which we call *subtopic structure*, is an important difference from informal chat/interpersonal banter. Figure 1 shows the subtopic structure and relation of the first 4 messages from a chat log, produced manually. Each message is represented horizontally; the vertical arrows show where participants responded to each other. Visual inspection reveals in this example there are three distinctive clusters (a more complex cluster and two smaller satellite clusters) of discussions between participants at sub-message level.

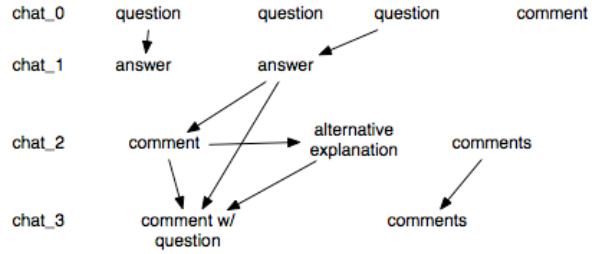

Figure 1. An example of chat subtopic structure and relation between correspondences.

3.4 Observation on Summary Digests

To measure the goodness of system-produced summaries, *gold standards* are used as references. Human-written summaries usually make up the gold standards. The Kernel Traffic (summary digests) are written by Linux experts who actively contribute to the production and discussion of the open source projects. However, participant-produced digests cannot be used as reference summaries verbatim. Due to the complex structure of the dialogue, the summary itself exhibits some discourse structure, necessitating such reader guidance phrases such as "for the … question," "on the … subject," "regarding …," "later in the same thread," etc., to direct and refocus the reader's attention. Therefore, further manual editing and partitioning is needed to transform a multi-topic digest into several smaller subtopic-based *gold-standard* reference summaries (see Section 6.1 for the transformation).

4 Fine-grained Clustering

To model the subtopic structure of each chat message, we apply clustering at the sub-message level.

4.1 Message Segmentation

First, we look at each message and assume that each participant responds to an ongoing discussion by stating his/her opinion on several topics or issues that have been discussed in the current chat log, but not necessarily in the order they were discussed. Thus, topic shifts can occur sequentially within a message. Messages are partitioned into multi-paragraph segments using TextTiling, which reportedly has an overall precision of 83% and recall of 78% (Hearst, 1994).

4.2 Clustering

After distinguishing a set of message segments, we cluster them. When choosing an appropriate clustering method, because the number of subtopics under discussion is unknown, we cannot make an assumption about the total number of resulting clusters. Thus, nonhierarchical partitioning methods cannot be used, and we must use a hierarchical method. These methods can be either *agglomerative*, which begin with an unclustered data set and perform $N - 1$ pairwise joins, or *divisive*, which add all objects to a single cluster, and then perform $N - 1$ divisions to create a hierarchy of smaller clusters, where N is the total number of items to be clustered (Frakes and Baeza-Yates, 1992).

Ward's Method

Hierarchical agglomerative clustering methods are commonly used and we employ Ward's method (Ward and Hook, 1963), in which the text segment pair merged at each stage is the one that minimizes the increase in total within-cluster variance.

Each cluster is represented by an L-dimensional vector $(x_{i1}, x_{i2}, \ldots, x_{iL})$ where each x_{ik} is the word's *tf • idf* score. If m_i is the number of objects in the cluster, the squared Euclidean distance between two segments i and j is:

$$d_{ij}^2 = \sum_{K=1}^{L} (x_{ik} - x_{jk})^2$$

When two segments are joined, the increase in variance I_{ij} is expressed as:

$$I_{ij} = \frac{m_i m_j}{m_i + m_j} d_{ij}^2$$

Number of Clusters

The process of joining clusters continues until the combination of any two clusters would destabilize the entire array of currently existing clusters produced from previous stages. At each stage, the two clusters x_{ik} and x_{jk} are chosen whose combination would cause the minimum increase in variance I_{ij}, expressed as a percentage of the variance change from the last round. If this percentage reaches a preset threshold, it means that the nearest two clusters are much further from each other compared to the previous round; therefore, joining of the two represents a destabilizing change, and should not take place.

Sub-message segments from resulting clusters are arranged according to the sequence the original messages were posted and the resulting subtopic structures are similar to the one shown in Figure 1.

5 Summary Extraction

Having obtained clusters of message segments focused on subtopics, we adopt the typical summarization paradigm to extract informative sentences and segments from each cluster to produce subtopic-based summaries. If a chat log has *n* clusters, then the corresponding summary will contain *n* mini-summaries.

All message segments in a cluster are related to the central topic, but to various degrees. Some are answers to questions asked previously, plus further elaborative explanations; some make suggestions and give advice where they are requested, etc. From careful analysis of the LKA data, we can safely assume that for this type of conversational interaction, the goal of the participants is to seek help or advice and advance their current knowledge on various technical subjects. This kind of interaction can be modeled as one problem-initiating segment and one or more corresponding problem-solving segments. We envisage that identifying corresponding message segment pairs will produce adequate summaries. This analysis follows the structural organization of summaries from Kernel Traffic. Other types of discussions, at least in part, require different discourse/summary organization.

These corresponding pairs are formally introduced below, and the methods we experimented with for identifying them are described.

5.1 Adjacent Response Pairs

An important conversational analysis concept, adjacent pairs (AP), is applied in our system to identify initiating and responding correspondences from different participants in one chat log. Adjacent pairs are considered fundamental units of conversational organization (Schegloff and Sacks, 1973). An adjacent pair is said to consist of two parts that are ordered, adjacent, and produced by different speakers (Galley et al., 2004). In our email/chat (LKA) corpus a physically adjacent message, following the timeline, may not directly respond to its immediate predecessor. Discussion participants read the current live thread and decide what he/she would like to correspond to, not necessarily in a serial fashion. With the added complication of subtopic structure (see Figure 1) the definition of adjacency is further violated. Due to its problematic nature, a relaxation on the adjacency requirement is used in extensive research in conversational analysis (Levinson, 1983). This relaxed requirement is adopted in our research.

Information produced by adjacent correspondences can be used to produce the subtopic-based summary of the chat log. As described in Section 4, each chat log is partitioned, at sub-message level, into several subtopic clusters. We take the message segment that appears first chronologically in the cluster as the topic-initiating segment in an adjacent pair. Given the initiating segment, we need to identify one or more segments from the same cluster that are the most direct and relevant responses. This process can be viewed equivalently as the informative sentence extraction process in conventional text-based summarization.

5.2 AP Corpus and Baseline

We manually tagged 100 chat logs for adjacent pairs. There are, on average, 11 messages per chat log and 3 segments per message (This is considerably larger than threads used in previous research). Each chat log has been clustered into one or more bags of message segments. The message segment that appears earliest in time in a cluster

was marked as the initiating segment. The annotators were provided with this segment and one other segment at a time, and were asked to decide whether the current message segment is a direct answer to the question asked, the suggestion that was requested, etc. in the initiating segment. There are 1521 adjacent response pairs; 1000 were used for training and 521 for testing.

Our baseline system selects the message segment (from a different author) immediately following the initiating segment. It is quite effective, with an accuracy of 64.67%. This is reasonable because not all adjacent responses are interrupted by messages responding to different earlier initiating messages.

In the following sections, we describe two machine learning methods that were used to identify the second element in an adjacent response pair and the features used for training. We view the problem as a binary classification problem, distinguishing less relevant responses from direct responses. Our approach is to assign a candidate message segment *c* an appropriate response class *r*.

5.3 Features

Structural and durational features have been demonstrated to improve performance significantly in conversational text analysis tasks. Using them, Galley et al. (2004) report an 8% increase in speaker identification. Zechner (2001) reports excellent results (F > .94) for inter-turn sentence boundary detection when recording the length of pause between utterances. In our corpus, durational information is nonexistent because chats and emails were mixed and no exact time recordings beside dates were reported. So we rely solely on structural and lexical features.

For structural features, we count the number of messages between the initiating message segment and the responding message segment. Lexical features are listed in Table 1. The tech words are the words that are uncommon in conventional literature and unique to Linux discussions.

5.4 Maximum Entropy

Maximum entropy has been proven to be an effective method in various natural language processing applications (Berger et al., 1996). For training and testing, we used YASMET[3]. To est imate $P(r \mid c)$ in the exponential form, we have:

- number of overlapping words
- number of overlapping content words
- ratio of overlapping words
- ratio of overlapping content words
- number of overlapping tech words

Table 1. Lexical features.

$$P_\lambda(r \mid c) = \frac{1}{Z_\lambda(c)} \exp(\sum_i \lambda_{i,r} f_{i,r}(c,r))$$

where $Z_\lambda(c)$ is a normalizing constant and the feature function for feature f_i and response class r is defined as:

$$f_{i,r}(c,r') = \begin{cases} 1, \text{ if } f_i > 0 \text{ and } r' = r \\ 0, \text{ otherwise} \end{cases} .$$

$\lambda_{i,r}$ is the feature-weight parameter for feature f_i and response class r. Then, to determine the best class r for the candidate message segment c, we have:

$$r^* = \arg\max_r P(r \mid c) \quad .$$

5.5 Support Vector Machine

Support vector machines (SVMs) have been shown to outperform other existing methods (naïve Bayes, k-NN, and decision trees) in text categorization (Joachims, 1998). Their advantages are robustness and the elimination of the need for feature selection and parameter tuning. SVMs find the hyperplane that separates the positive and negative training examples with maximum margin. Finding this hyperplane can be translated into an optimization problem of finding a set of coefficients α_i^* of the weight vector $\vec{w}$ for document d_i of class $y_i \in \{+1, -1\}$:

$$\vec{w} = \sum_i \alpha_i^* y_i \vec{d}_i, \quad \alpha_i > 0 \quad .$$

Testing data are classified depending on the side of the hyperplane they fall on. We used the LIBSVM[4] package for training and testing.

Feature sets	baseline	MaxEnt	SVM
	64.67%		
Structural		61.22%	71.79%
Lexical		62.24%	72.22%
Structural + Lexical		72.61%	72.79%

Table 2. Accuracy on identifying APs.

[3] http://www.fjoch.com/YASMET.html
[4] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

5.6 Results

Entries in Table 2 show the accuracies achieved using machine learning models and feature sets.

5.7 Summary Generation

After responding message segments are identified, we couple them with their respective initiating segment to form a mini-summary based on their subtopic. Each initializing segment has zero or more responding segments. We also observed *zero response* in human-written summaries where participants initiated some question or concern, but others failed to follow up on the discussion. The AP process is repeated for each cluster created previously. One or more subtopic-based mini-summaries make up one final summary for each chat log. Figure 2 shows an example. For longer chat logs, the length of the final summary is arbitrarily averaged at 35% of the original.

Subtopic 1:
Benjamin Reed: I wrote a wireless ethernet driver a while ago... Are driver writers recommended to use that over extending **/proc** or is it deprecated?
Linus Torvalds: **Syscyl** is deprecated. It's useful in one way only ...

Subtopic 2:
Benjamin Reed: I am a bit uncomfortable ... wondering for a while if there are **guidelines** on …
Linus Torvalds: The thing to do is to create ...

Subtopic 3:
Marcin Dalecki: Are you just blind to the never-ending **format/ compatibility/ … problems** the whole idea behind /proc induces inherently?

Figure 2. A system-produced summary.

6 Summary Evaluation

To evaluate the goodness of the system-produced summaries, a set of reference summaries is used for comparison. In this section, we describe the manual procedure used to produce the reference summaries, and the performances of our system and two baseline systems.

6.1 Reference Summaries

Kernel Traffic digests are participant-written summaries of the chat logs. Each digest mixes the summary writer's own narrative comments with direct quotes (citing the authors) from the chat log. As observed in Section 3.4, subtopics are intermingled in each digest. Authors use key phrases to link the contents of each subtopic throughout texts. In Figure 3, we show an example of such a digest. Discussion participants' names are in italics and subtopics are in bold. In this example, the conversation was started by Benjamin Reed with two questions: 1) asking for conventions for writing /proc drivers, and 2) asking about the status of sysctl. The summary writer indicated that Linus Torvalds replied to both questions and used the phrase "for the … question, he added…" to highlight the answer to the second question. As the di-

Benjamin Reed wrote a wireless Ethernet **driver that used /proc** as its interface. But he was a little uncomfortable … asked if there were any conventions he should follow. He added, "and finally, what's up with **sysctl**? …"
Linus Torvalds replied with: "the thing to do is to create a …[program code]. The **/proc/drivers/** directory is already there, so you'd basically do something like … [program code]." For the **sysctl** question, he added "sysctl is deprecated. ..."
Marcin Dalecki flamed Linus: "Are you just blind to the never-ending format/compatibility/… problems the whole idea behind **/proc** induces inherently? …[example]"

Figure 3. An original Kernel Traffic digest.

Mini 1:
Benjamin Reed wrote a wireless Ethernet **driver that used /proc** as its interface. But he was a little uncomfortable … and asked if there were any conventions he should follow.
Linus Torvalds replied with: the thing to do is to create a …[program code]. The **/proc/drivers/** directory is already there, so you'd basically do something like … [program code].
Marcin Dalecki flamed Linus: Are you just blind to the never-ending format/ compatibility/ … problems the whole idea behind **/proc** induces inherently? …[example]

Mini 2:
Benjamin Reed: and finally, what's up with **sysctl**? ...
Linus Torvalds replied: sysctl is deprecated. ...

Figure 4. A reference summary reproduced from a summary digest.

gest goes on, Marcin Dalecki only responded to the first question with his excited commentary.

Since our system-produced summaries are subtopic-based and partitioned accordingly, if we use unprocessed Kernel Traffic as references, the comparison would be rather complicated and would increase the level of inconsistency in future assessments. We manually reorganized each summary digest into one or more mini-summaries by subtopic (see Figure 4.) Examples (usually kernel stats) and programs are reduced to "[example]" and "[program code]." Quotes (originally in separate messages but merged by the summary writer) that contain multiple topics are segmented and the participant's name is inserted for each segment. We follow clues like "to answer … question" to pair up the main topics and their responses.

6.2 Summarization Results

We evaluated 10 chat logs. On average, each contains approximately 50 multi-paragraph tiles (partitioned by TextTile) and 5 subtopics (clustered by the method from Section 4).

A simple baseline system takes the first sentence from each email in the sequence that they were posted, based on the assumption that people tend to put important information in the beginning of texts (*Position Hypothesis*).

A second baseline system was built based on constructing and analyzing the dialogue structure of each chat log. Participants often quote portions of previously posted messages in their responses. These quotes link most of the messages from a chat log. The message segment that immediately follows the quote is automatically paired with the quote itself and added to the summary and sorted according to the timeline. Segments that are not quoted in later messages are labeled as less relevant and discarded. A resulting baseline summary is an inter-connected structure of segments that quoted and responded to one another. Figure 5 is a shortened summary produced by this baseline for the ongoing example.

[0|0] *Benjamin Reed*: "I wrote an … driver … /proc …"
[0|1] *Benjamin Reed*: "… /proc/ guideline …"
[0|2] *Benjamin Reed*: "… syscyl …"
[1|0] *Linus Torvalds* responds to [0|0, 0|1, 0|2]: "the thing to do is …" "sysctl is deprecated … "

Figure 5. A short example from Baseline 2.

The summary digests from Kernel Traffic mostly consist of direct snippets from original messages, thus making the reference summaries extractive even after rewriting. This makes it possible to conduct an automatic evaluation. A computerized procedure calculates the overlap between reference and system-produced summary units. Since each system-produced summary is a set of mini-summaries based on subtopics, we also compared the subtopics against those appearing in reference summaries (precision = 77.00%, recall = 74.33 %, F = 0.7566).

		Recall	Precision	F-measure
Baseline1		30.79%	16.81%	.2175
Baseline2		63.14%	36.54%	.4629
System	Summary	52.57%	52.14%	.5235
	Topic-summ	52.57%	63.66%	.5758

Table 3. Summary of results.

Table 3 shows the *recall*, *precision*, and *F-measure* from the evaluation. From manual analysis on the results, we notice that the original digest writers often leave large portions of the discussion out and focus on a few topics. We think this is because among the participants, some are Linux veterans and others are novice programmers. Digest writers recognize this difference and reflect it in their writings, whereas our system does not. The entry "*Topic-summ*" in the table shows system-produced summaries being compared only against the topics discussed in the reference summaries.

6.3 Discussion

A recall of 30.79% from the simple baseline reassures us the Position Hypothesis still applies in conversational discussions. The second baseline performs extremely well on recall, 63.14%. It shows that quoted message segments, and thereby derived dialogue structure, are quite indicative of where the important information resides. Systems built on these properties are good summarization systems and hard-to-beat baselines. The system described in this paper (*Summary*) shows an *F-measure* of .5235, an improvement from .4629 of the smart baseline. It gains from a high precision because less relevant message segments are identified and excluded from the adjacent response pairs,

leaving mostly topic-oriented segments in summaries. There is a slight improvement when assessing against only those subtopics appeared in the reference summaries (*Topic-summ*). This shows that we only identified clusters on their information content, not on their respective writers' experience and reliability of knowledge.

In the original summary digests, interactions and reactions between participants are sometimes described. Digest writers insert terms like "flamed", "surprised", "felt sorry", "excited", etc. To analyze social and organizational culture in a virtual environment, we need not only information extracts (implemented so far) but also passages that reveal the personal aspect of the communications. We plan to incorporate opinion identification into the current system in the future.

7 Conclusion and Future Work

In this paper we have described a system that performs intra-message topic-based summarization by clustering message segments and classifying topic-initiating and responding pairs. Our approach is an initial step in developing a framework that can eventually reflect the human interactions in virtual environments. In future work, we need to prioritize information according to the perceived knowledgeability of each participant in the discussion, in addition to identifying informative content and recognizing dialogue structure. While the approach to the detection of initiating-responding pairs is quite effective, differentiating important and non-important topic clusters is still unresolved and must be explored.

References

M. S. Ackerman and C. Halverson. 2000. Reexaming organizational memory. *Communications of the ACM*, 43(1), 59–64.

A. Berger, S. Della Pietra, and V. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

M. Elliott and W. Scacchi. 2004. Free software development: cooperation and conflict in a virtual organizational culture. S. Koch (ed.), *Free/Open Source Software Development*, IDEA publishing, 2004.

W. B. Frakes and R. Baeza-Yates. 1992. Information retrieval: data structures & algorithms. Prentice Hall.

M. Galley, K. McKeown, J. Hirschberg, and E. Shriberg. 2004. Identifying agreement and disagreement in conversational speech: use of Bayesian networks to model pragmatic dependencies. In the *Proceedings of ACL-04*.

M. A. Hearst. 1994. Multi-paragraph segmentation of expository text. In the *Proceedings of ACL 1994*.

T. Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the ECML*, pages 137–142.

D. Lam and S. L. Rohall. 2002. Exploiting e-mail structure to improve summarization. *Technical Paper at IBM Watson Research Center* #20–02.

S. Levinson. 1983. *Pragmatics*. Cambridge University Press.

P. Newman and J. Blitzer. 2002. Summarizing archived discussions: a beginning. In *Proceedings of Intelligent User Interfaces*.

O. Rambow, L. Shrestha, J. Chen and C. Laurdisen. 2004. Summarizing email threads. In *Proceedings of HLT-NAACL 2004: Short Papers*.

K. Ries. 2001. Segmenting conversations by topic, initiative, and style. In *Proceedings of SIGIR Workshop: Information Retrieval Techniques for Speech Applications 2001*: 51–66.

E. A. Schegloff and H. Sacks. 1973. Opening up closings. *Semiotica*, 7-4:289–327.

S. Wan and K. McKeown. 2004. Generating overview summaries of ongoing email thread discussions. In *Proceedings of COLING 2004*.

J. H. Ward Jr. and M. E. Hook. 1963. Application of an hierarchical grouping procedure to a problem of grouping profiles. *Educational and Psychological Measurement*, 23, 69–81.

K. Zechner. 2001. Automatic generation of concise summaries of spoken dialogues in unrestricted domains. In *Proceedings of SIGIR 2001*.